

What is claimed is:

1. A method for analyzing a program, comprising:

determining a set of functions required by the program by performing

5 local type constraint analysis at intermediate language instruction level and a call path
that may reach a function containing such instruction.

2. The method of Claim 1, further comprising:

analyzing a program instruction that accesses an object field, wherein the
analysis is performed locally to an object instantiation.

10 3. The method of Claim 1, further comprising:

analyzing a program instruction that accesses an array element locally to
an array instantiation.

4. The method of Claim 1, further comprising:

analyzing a program instruction that accesses runtime information for a
15 local runtime symbol usage.

5. The method of Claim 1, further comprising:

analyzing a program instruction within an exception handler performed
locally to an exception instruction.

6. The method of Claim 1, further comprising:

20 declaring possible return types of native functions, where a type analysis
of intermediate language instruction is not possible.

7. The method of Claim 6, wherein the set of functions may be in a single program
image.

8. A computer-readable medium storing computer-executable process steps of a process for analyzing a program, comprising:

determining a set of functions required by the program by performing local type constraint analysis at intermediate language instruction level and a call path that may reach a function containing such instruction.

9. The computer readable medium of Claim 8, further comprising:

analyzing a program instruction that accesses an object field, wherein the analysis is performed locally to an object instantiation.

10. The computer readable medium of Claim 8, further comprising:

analyzing a program instruction that accesses an array element locally to an array instantiation.

11. The computer readable medium of Claim 8, further comprising:

analyzing a program instruction that accesses runtime information for a local runtime symbol usage.

12. The computer readable medium of Claim 8, further comprising:

analyzing a program instruction within an exception handler performed locally to an exception instruction.

13. The computer readable medium of Claim 8, further comprising:

declaring possible return types of native functions, where a type analysis of intermediate language instruction is not possible.

14. The computer readable medium of Claim 13, wherein the set of functions may be in a single program image.

15. A method for analyzing a program, comprising:

determining an object type that may exist at an execution point of the program, wherein this enables determination of a possible virtual function that may be called.

16. The method of Claim 15, further comprising:

- 5 creating a call graph at a main entry point of the program; and
 recording an outgoing function call within a main function.

17. The method of Claim 16, further comprising:

 analyzing possible object types that may occur at any given instruction from any call path for a virtual call.

- 10 18. The method of Claim 17, wherein possible object types are determined by tracking object types as they pass through plural constructs.

19. The method of Claim 15, further comprising:

 calling into function generically for handling specialized native runtime type information.

- 15 20. A computer-readable medium storing computer-executable process steps of a process for analyzing a program, comprising:

 determining an object type that may exist at an execution point of the program, wherein this enables determination of possible virtual functions that may be called.

- 20 21. The computer readable medium of Claim 20, further comprising:

 creating a call graph at a main entry point of the program; and
 recording an outgoing function call within a main function.

22. The computer readable medium of Claim 21 further comprising:

analyzing possible object types that may occur at any given instruction
from a call path for virtual calls.

23. The computer readable medium of Claim 22, wherein possible object types are
determined by tracking object types as they pass through plural constructs.

5 24. The computer readable medium of Claim 20, further comprising:

calling into functions generically for handling specialized native runtime
type information.

25. A method for building an application, comprising:

receiving source code instruction;

10 determining optimum code requirement; and

compiling native processor image.

26. The method of Claim 25, wherein the optimum code is determined by performing a
flow-sensitive analysis that determines possible types of objects that may exist at any
instruction of a program.

15 27. The method of Claim 26, wherein based on a set of constraints, virtual functions that
have the potential of being executed are determined.

28. A computer-readable medium storing computer-executable process steps of a process
for building an application, comprising:

receiving source code instruction;

20 determining optimum code requirement; and

compiling native processor image.

29. The computer readable medium of Claim 28, wherein the optimum code is determined by performing a flow-sensitive analysis that determines possible types of objects that may exist at any instruction of a program.

30. The computer readable medium of Claim 29, wherein based on a set of constraints,
5 virtual functions that have the potential of being executed are determined.

31. The method of Claim 1, wherein the program runs in a managed runtime environment.

32. The computer readable medium of Claim 8, wherein the program runs in a managed runtime environment.

10 33. The method of Claim 15, wherein the program runs in a managed runtime environment.

34. The computer readable medium of Claim 20, wherein the program runs in a managed runtime environment.

35. The method of Claim 25, wherein the program runs in a managed runtime
15 environment.

36. The computer readable medium of Claim 28, wherein the program runs in a managed runtime environment.

37. A method for determining variable size in a program, comprising:

tracking variable size; and

20 reducing variable size for program execution.

38. The method of Claim 37, wherein if a variable is discrete, then it is hard coded to a single value.

39. The method of Claim 37, wherein if a first variable is assigned to a second variable, then a size constraint of the first variable is merged into a size constraint of the second variable.

40. A computer-readable medium storing computer-executable process steps of a process
5 for determining variable size in a program, comprising:
tracking variable size; and
reducing variable size for program execution.

41. The computer readable medium of Claim 40, wherein if a variable is discrete, then it is hard coded to a single value.

10 42. The computer readable medium of Claim 40, wherein if a first variable is assigned to a second variable, then a size constraint of the first variable is merged into a size constraint of the second variable.

43. A method for reducing empty function calls in a program, comprising:
determining if a call is made to an empty function; and
15 removing a call that is made to an empty function.

44. A computer-readable medium storing computer-executable process steps of a process for reducing empty function calls in a program, comprising:

determining if a call is made to an empty function; and
removing a call that is made to an empty function.

20 45. A method for reducing throw instruction without exception handlers in a program, comprising:

determining if there are any throw instructions without exception handlers;
and

removing throw instructions without exception handlers.

46. A computer-readable medium storing computer-executable process steps of a process for reducing throw instruction without exception handlers in a program, comprising:

determining if there are any throw instructions without exception handlers;

5 and

removing throw instructions without exception handlers.

47. A method for discarding comparison instructions in a program, comprising:

determining if there are any comparison instructions with discrete values
in the program;

10 discarding a comparison instruction with a discrete value.

48. A computer-readable medium storing computer-executable process steps of a process for discarding comparison instructions in a program, comprising:

determining if there are any comparison instructions with discrete values
in the program; and

15 discarding a comparison instruction with a discrete value.